# Software Design Specification Document

STORC DASHBOARD PROJECT

CD Jam

CSC 191 – FALL 2015 | 10/23/2015

# Table of Contents

1. INTRODUCTION ... 4
1.1 Purpose ... 4
1.2 Scope ... 5
1.3 Definitions, Acronyms, and Abbreviations ... 5
1.3.1 Definitions ... 5
1.3.2 Acronyms ... 5
1.3.3 Abbreviations ... 6
1.4 References ... 6
1.5 Overview of Contents of Document ... 6
2. ARCHITECTURAL DESIGN ... 6
2.1 Hardware Architecture ... 6
2.2 Software Design Architecture ... 8
2.2.1 Presentation Layer ... 9
2.2.2 "Business" Logic Layer ... 9
2.2.3 Date Management Layer ... 9
3. INTERFACE DESIGN ... 9
3.1 Feature 1: Homepage Management Interface ... 12
3.2 Feature 2: Project Selection Interface ... 12
3.3 Feature 3: Date Input Interface ... 13
3.4 Feature 4: Review and Submit Data Interface ... 13
3.5 Feature 5: Add/Remove/Edit Users Interface ... 13
3.6 Feature 6: View/Select Pending Data ... 14
3.7 Feature 7: Select/Edit Project of Station Interface ... 14
4. DATABASE SCHEMA ... 15
4.1 ERD diagram ... 15
4.2 Creating the database ... 16
5. COMPONENT DESIGN SPECIFICATIONS ... 20
5.1 Sequence Diagrams (Introduction) ... 21
5.1.1 Component 1: Add User Sequence Diagram ... 21
5.1.2 Component 2: Change User Permissions Sequence Diagram ... 21
5.1.3 Component 3: Data Input Sequence Diagram ... 22

5.1.4 Component 4: Edit Data Sequence Diagram ............ 22
5.1.5 Component 5: Remove Data Sequence Diagram ............ 23
5.1.6 Component 6: Login Sequence Diagram ............ 23
5.1.7 Component 7: Report Data Sequence Diagram ............ 23
5.1.8 Component 8: Remove User Sequence Diagram ............ 24
5.1.9 Component 9: Create New Project Sequence Diagram ............ 24
5.1.10 Component 10: Add Widget Sequence Diagram ............ 25
5.1.11 Component 11: Remove Widget Sequence Diagram ............ 25
5.1.12 Edit Widget ............ 26
5.2 Webpage and Function Design Specifications ............ 26
5.2.1 Dashboard.apx ............ 26
5.2.2 Dashboard.cs ............ 26
5.2.3 ManageWid.aspx ............ 26
5.2.4 ManageWid.cs ............ 26
5.2.5 ManageUsers.aspx ............ 27
5.2.6 ManageUsers.cs ............ 27
5.2.7 InputData.aspx ............ 27
5.2.8 InputData.cs ............ 27
5.2.9 ManageData.aspx ............ 27
5.2.10 ManageData.cs ............ 28
5.2.11 Login.html ............ 28
6. PERFORMANCE ANALYSIS ............ 28
7. RESOURCE ESTIMATES ............ 28
8. SOFTWARE REQUIREMENTS TRACEABLITY MATRIX ............ 28
3.1 Homepage Management Interface ............ 28
3.3 Data Input Interface, ............ 28
3.2 Project Selection Interface ............ 28
9. APPROVALS ............ 29
APPENDIX A: Database tables and attributes ............ 30
Access List Table ............ 30
Biodiesel Station Table ............ 30
Biodiesel DataPoint Table ............ 31

Domain Table 33
Energy Consumption DataPoint Table 33
Fishtank_Stations Table 34
Fishtank DataPoint Table 34
Growbed Stations Table 35
Growbed DataPoint Table 35
Projects Table 36
Solar Stations table 37
Solar DataPoint Table 38
Wormbin Stations Table 39
Wormbin DataPoint Table 40
Users Table 41
APPENDIX B: Alphabetic listing of each attribute and its characteristics 42
APPENDIX C: Coding Standards 45

# 1. INTRODUCTION

This is the Software Design Specification document for the STORC Dashboard Project sponsored by Dr. Michael Christensen.

This project is being undertaken by the CD Jam software development team. The team is comprised of undergraduate students majoring in Computer Science at California State University, Sacramento. The team members are enrolled in a two-semester senior project course required of all undergraduate majors. Successful delivery of the desired software product will fulfill the senior project requirement for the student team members.


PROJECT SPONSOR:
*Name*: Michael Christensen
*Title:* Assistant Vice President for Risk Management Services & Director of STORC
*Organization Name:* Sustainability Technology Optimization Research Center (STORC)
*Contact Information:*
*Phone:* (916) 278-5252
*Email:* storc@csus.edu

CD JAM DEVELOPMENT TEAM:
*Name:* Cole Culler
*Contact Information:*
*Phone:* (530) 575-7683
*Email:* cbroski.culler@gmail.com
*Name:* David Grapentine
*Contact Information:*
*Phone:* (707) 471-8749
*Email:* davidjoaograpentine@gmail.com
*Name:* Ashley Gregory
*Contact Information:*
*Phone:* (209) 304-1884
*Email:* aa5gregory@gmail.com
*Name:* John Jones
*Contact Information:*
*Phone:* (916) 475-8460
*Email:* felixequal@gmail.com
*Name:* Michael Smith
*Contact Information:*
*Phone:* (916) 842-8339
*Email:* infinitlyill@gmail.com


## 1.1 Purpose

The purpose of the Software Design Specification is to describe the components, graphic user interface, and data within in the STORC Dashboard Project. This document includes diagrams, charts, and models exhibiting the interactions between STORC users, STORC/IRT Servers, and web pages. This document will also reiterate some of the aspects seen in the SRS and past documents.

## 1.2 Scope

The SDS will cover the components, graphic user interface, and data implemented in the STORC Dashboard Project. Other documents like the STS and SRS will explain other relationship this system has with itself and the users. Any and all schedule changes need to be made by the team as a whole. This process is explained in more depth in the Software Project Management Plan.

## 1.3 Definitions, Acronyms, and Abbreviations

### 1.3.1 Definitions

**Administrator:** A super user that oversees other users and the projects within STORC.
**Aquaponics:** A cycle between hydroponically grown plants and aquatic animals, in which the waste produced from animals supplies nutrients for plants which in turn purifies the water.
**Biodiesel:** A substitute for diesel created by a biological chemical reaction.
**Dashboard:** A collection of data laid out in an easy to read format represented in a graphical format.
**Photovoltaic Cell:** A device that delivers an electric current as a result of a chemical reaction from the rays of the sun.
**Principal Investigator:** A user that in charge one or more STORC projects and oversees one or more STORC Technicians. Principal Investigators are most often faculty or staff members at CSUS.
**Public:** Any user that is not directly involved with STORC or STORC activities.
**Technician:** A user who works on a project overseen by a PI. This user generally monitors and collects data directly from one or more project stations. These are usually student volunteers.
**Vermiculture:** The cultivation of worms used for composting materials.
**Webmaster:** This user is in charge of maintaining and updating the default web page and widgets for the STORC Dashboard Project.

### 1.3.2 Acronyms

**CSc –** Computer Science
**CSUS** – California State University, Sacramento
**ECS –** College of Engineering and Computer Science
**ERD** – Entity Relationship Diagram
**GUI** – Graphical User Interface
**HTML –** Hyper Text Markup Language
**IRT –** Information Resources and Technology
**IT –** Information Technology
**MySQL –** My Structured Query Language
**PI –** Principal Investigator
**PMP –** Software Project Management Plan
**SDS –** Software Design Specification
**SRS –** Software Requirements

**STR –** System Test Report
**STORC –** Sustainability Technology Optimization Research Center
**STS -** Software Test Specification
**UM –** User Manual
**UML –** Unified Modeling Language
**WCM** – Web Content Management

### 1.3.3 Abbreviations

**Admin** – Administrator
**CSc 190:** Computer Science Senior Project - Part 1
**CSc 191:** Computer Science Senior Project - Part 2
**Tech** – Technician

## 1.4 References

Buckley, Bob. *CSc 190-01 Senior Project: Part 1.* CSUS, Dec. 2014. Web. 22 February 2015. http://athena.ecs.csus.edu/~buckley/CSc190/CSc190.html

*STORC*. CSUS STORC. n.p. Web. 22 February 2015. http://www.csus.edu/storc/about.html

## 1.5 Overview of Contents of Document

There are seven different sections comprising the SDS: Architectural Design, Interface Design, Database Schema, Component Design Specifications, Performance Analysis, Resource Estimates, and the Software Requirement Traceability Matrix. *Section 2: Architectural Design* shows the relationship between each widget and the data being displayed based on the user's selections. *Section 3: Interface Design* gives an overview of the look and feel of the STORC Dashboard Project showed with prototypes of widgets and web pages. The *Section 4: Database Schema* shows the layout of the database and how to create new instances of projects, stations, and other related data. *Section 5: Component Design Specifications* explains the relationship between an action performed by the user and the files associated with these actions. *Section 6: Performance Analysis* introduces any issues the STORC Dashboard project may encounter with respects to performance. *Section 7: Resource Estimates* section discusses the cost of resources needed to setup, run and maintain the software. *Section 8: Software Requirements Traceability Matrix* shows and relates each function in the STORC Dashboard Projects to subsections in the SRS.

# 2. ARCHITECTURAL DESIGN

## 2.1 Hardware Architecture

The STORC Dashboard utilizes a n-tiered architecture to structure the hardware aspect of the project. The starting point is users device; which may be any electronic device that can access the internet. This device uses the internet in order to navigate to the STORC website where users can access the STORC Dashboard Project web interface (Number 1 in Figure 1). Authorized users will be able to access the STORC Dashboard Project web interface by requesting login information from IRT Servers (Number 2 in Figure 1). Here the IRT Servers will authenticate the login information of the user. If correct the IRT Server will grant them access to get information stored STORC Database(Number 5 in Figure 1) . Once granted (Number 6 in Figure 1), the STORC Server will then populate the users homepage (Number 7 in Figure 1). Once the homepage has loaded completely the user will be able to use the features of the STORC Dashboard Project (Number 8 in Figure 1).

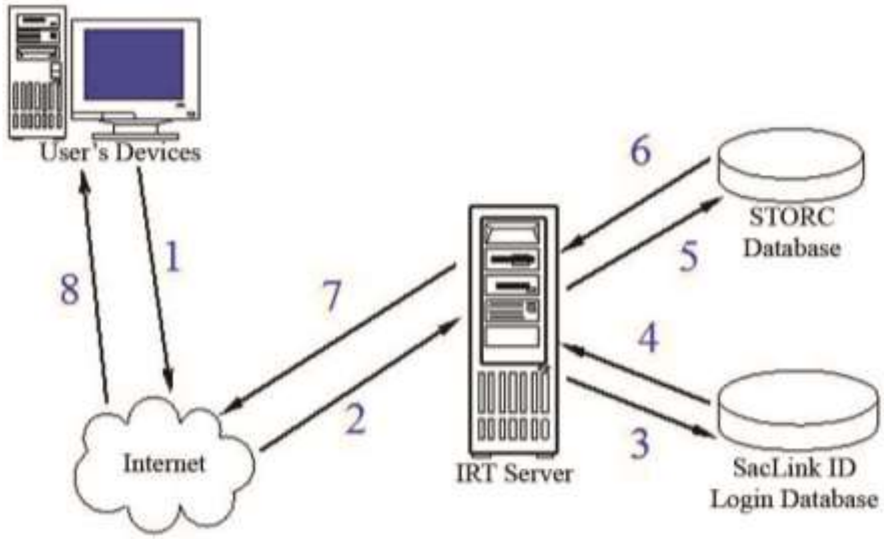


*Figure 1: Hardware Architecture – System*

The STORC Dashboard Project will incorporate sensors that report vital information about projects being conducted at STORC. This data will be stored in the STORC Database (Number 1 in Figure 2). The STORC Server will sync with an IRT Server (Number 2 in Figure 2) and transfer the data to the STORC Database, which is located on the IRT Server (Number 3 in Figure 2).

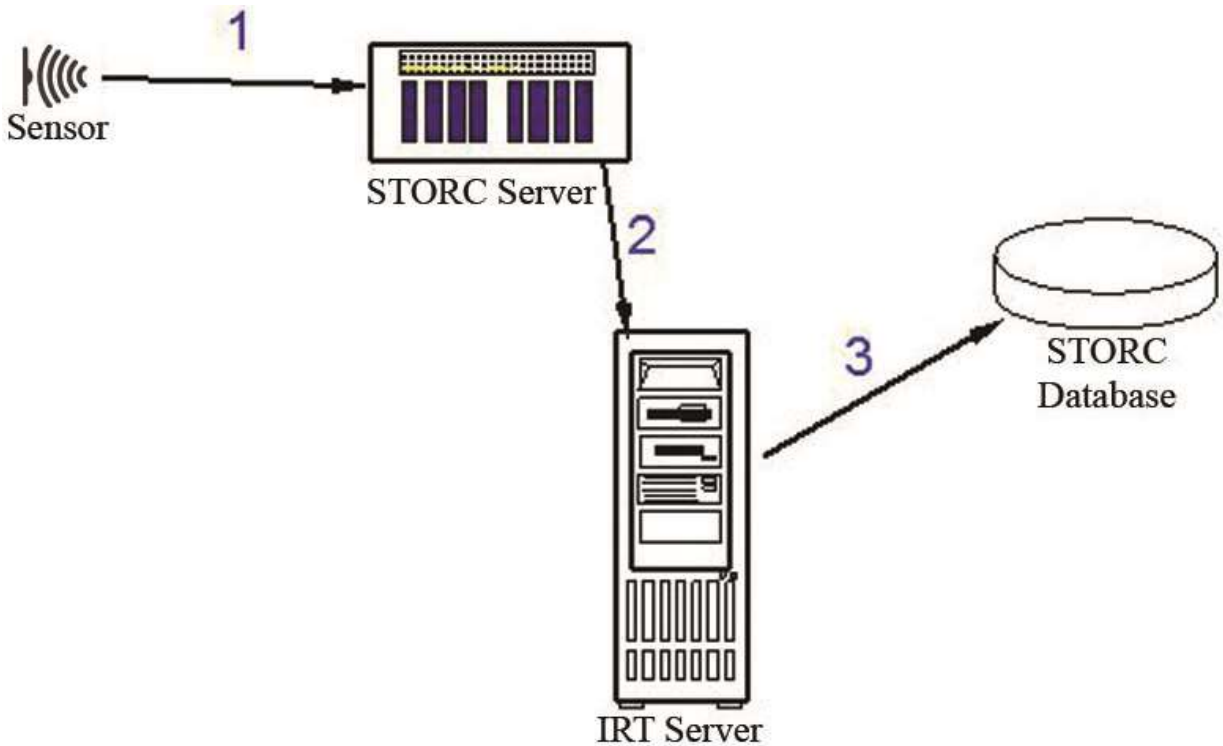

*Figure 2: Hardware Architecture - Sensor*

## 2.2 Software Design Architecture

The STORC Dashboard project is based of a simple MVC model (See Figure 3).

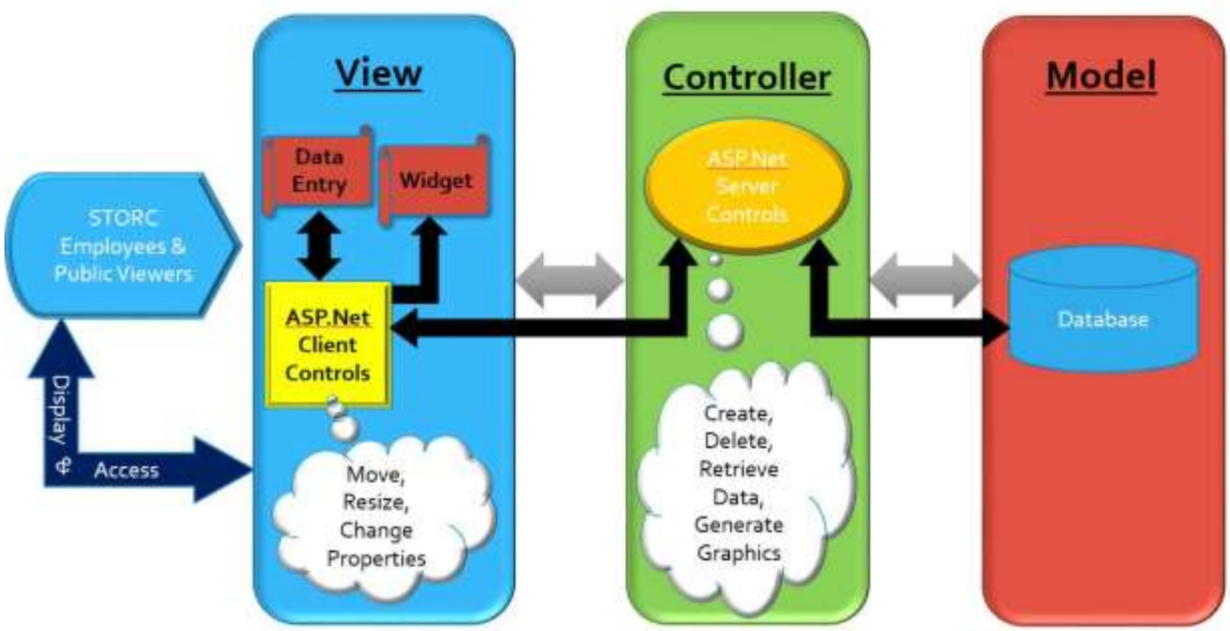

*Figure 3: Software Design Architecture*

### 2.2.1 Presentation Layer

In order to present the data STORC collects to the public, we will be using various forms of widgets that will be displayed on a public website. STORC employees such as PI's and Tech's will have access to a customizable website where they can change the data displayed by the widgets to meet their needs. There will be many types of widgets that will be used to display the data including, pie charts, histograms, bar graphs, and spreadsheets. Each STORC Employee will be able to customize their own webpage, in which they can layout the series of widgets to suit each of their individual needs. To manage the data collected, users have access to change the information stored in the database in case a mistake was made or a sensor become faulty. In order to do this the user will use a GUI customized to only change the data for the projects are working on.

### 2.2.2 "Business" Logic Layer

Our team is using the ASP.NET framework which utilizes C# and Javascript to implement the business rules set forth by our sponsor. Since the software is web-form based the Business Layer bridges the web-form and the MSSql database. This layer is written in C# in order to utilize the power of the language and to create customizable widgets which can be modified in real time.

### 2.2.3 Date Management Layer

We will be taking advantage of the fact that the ASP.NET framework has built in database management controls, compatible with a variety of database management systems. Our DBMS will be Microsoft SQL Server. The ASP.NET Server Controls within the ASP.NET framework will be able to create, remove, update, and delete data with in the database, when users enter or edit the information that has been collected on-site at STORC. Users will manipulate this data via the data entry view mentioned previously in the presentation layer. The Microsoft SQL server database will be maintained on campus by the Information Resource and Technology department.

# 3. INTERFACE DESIGN

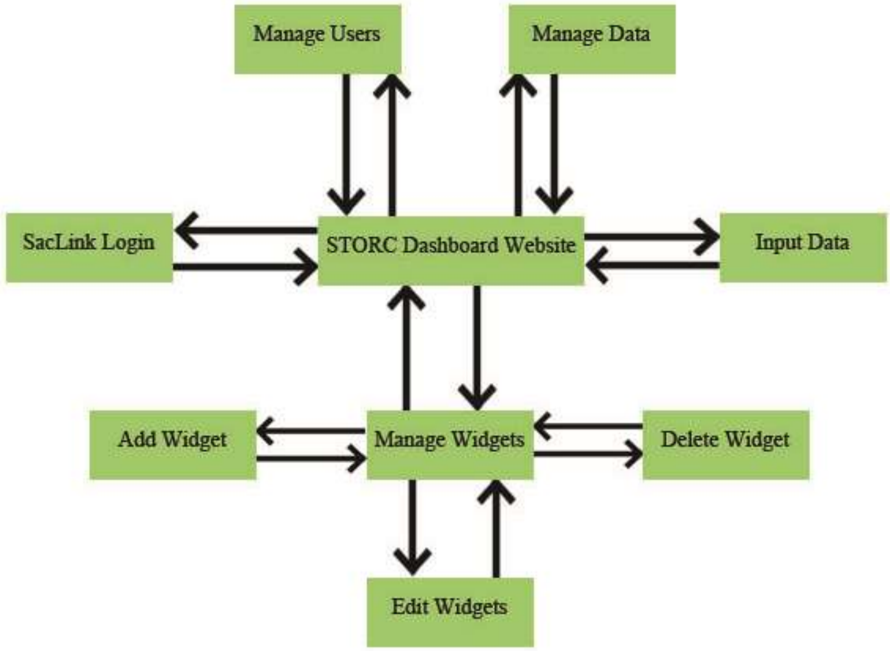


*Figure 4: Interface Design*

| Feature | Description | Webpages |
|---|---|---|
| Homepage Management Interface | 1.Customize Widget - Right clicking on one of the widgets, the user will be presented with menus and dropdowns that can change the widget's data source(s), widget's chart type, colors, and other relevant settings.<br>2. Delete Widget - Clicking the X-icon in the top right corner of the widget, the widget can be deleted. There will most likely be a prompt to verify the user's decision before this widget deletion.<br>3. Add New Widget - By clicking the Add New Widget button, the user will be prompted for widget data sources, chart type, color, and other relevant settings. After configuring the new widget's settings, a new widget will be added to the dashboard which the user can position and resize as they see fit.<br>4. Move/Resize Widget - Moving and resizing a widget could be as simple as selecting the upper right corner of the widget and dragging and panning to resize and situate the widget on the dashboard. It is likely that automatic snap functionality will be included to make this supporting feature more user-friendly.<br>5.Create New Project Tab | Dashboard.html,<br>Dashboard.cs,<br>Dashboard.js,<br>Dashbard.aspx<br><br>ManageWid.html,<br>ManageWid.cs,<br>ManageWid.js,<br>ManageWid.aspx<br><br>EditWid.html,<br>EditWid.cs,<br>EditWid.js,<br>EditWid.aspx<br><br>DeleteWid.html,<br>DeleteWid.cs,<br>DeleteWid.js,<br>DeleteWid.aspx<br><br>ManageData.html,<br>ManageData.cs,<br>ManageData.js,<br>ManageData.aspx<br><br>ManageUsers.html, |

| | | |
|---|---|---|
| | | ManageUsers.cs,<br>ManageUsers.js,<br>ManageUsers.aspx |
| Project Selection Interface | 1. Select Project and Station - Users will select individual projects and stations that require manual input of data. Once selected they can proceed to the input form. | ManageData.html,<br>ManageData.cs,<br>ManageData.js,<br>ManageData.aspx<br><br>ManageUsers.html,<br>ManageUsers.cs,<br>ManageUsers.js,<br>ManageUsers.aspx |
| Data Input Interface | 1. The user will be presented with a single row to fill out. This row will automatically show the column categories to make it easier to correlate a user's written data with the data to be entered. Once a row has been filled, a new row could automatically be generated to improve data entry flow. | InputData.html,<br>InputData.cs,<br>InputData.js,<br>InputData.aspx |
| Review and Submit Data Interface | Once the user has finished filling out the input form, they will be able to review all data they have inputted and then submit the data to the Principal Investigator for approval. | InputData.html,<br>InputData.cs,<br>InputData.js,<br>InputData.aspx |
| Add/Remove/ Edit Users Interface | 1. Select between Add User, Delete User, Edit Permissions - A list of all current users could be filtered and added to an edit list. All users on the edit list would then be affected by any changes the admin or PI selects.<br>2. Add or Remove Permissions or Access - A series of dropdown boxes and buttons could add or remove access to each user on the edit list. These changes would be shown in the list on the right for clarity.<br>3. Undo Changes | ManageUsers.html,<br>ManageUsers.cs,<br>ManageUsers.js,<br>ManageUsers.aspx |
| View/Select Pending Data | 1. View and Select Pending Data - By clicking on a particular user submission, the data for that submission could be viewed beneath in a separate window. Once this data has been checked for errors and committed, a check mark could indicate that this particular technician's submission has been completed. The Admin or PI could then move on to the next item.<br>2. Edit, Reject, or Commit | InputData.html,<br>InputData.cs,<br>InputData.js,<br>InputData.aspx |
| Select/Edit Project or Station Interface | 1. Select Project and Station to Edit - An Admin or PI could directly correct erroneous data in the database by selecting a particular data set from a project and station. This would populate the data below in a simple to read table.<br>2. Edit Data - Admins or PIs could edit any field by using a simple click and edit interface, much like Excel. Once | ManageData.html,<br>ManageData.cs,<br>ManageData.js,<br>ManageData.aspx |

| | | |
|---|---|---|
| | the data is correct, the Admin or PI could commit this data back into the database. | |

### 3.1 Feature 1: Homepage Management Interface

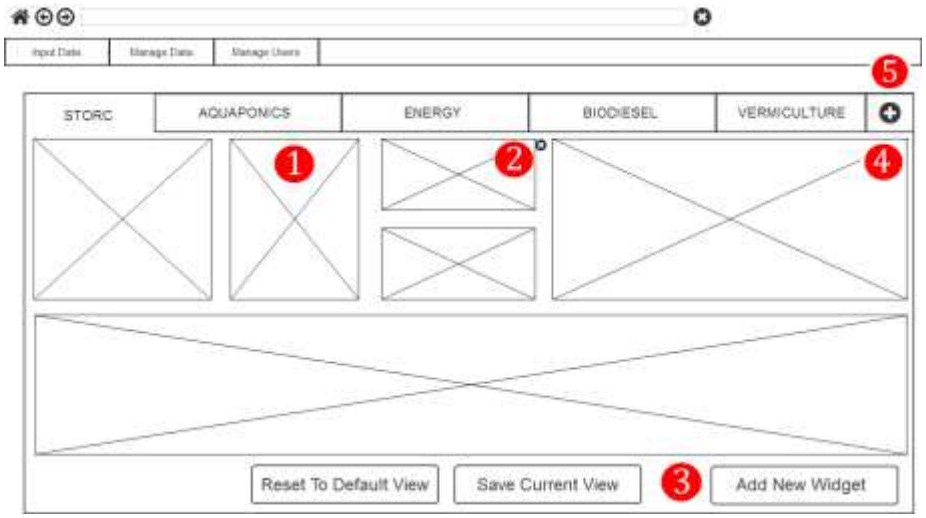


*Figure 5: Homepage Management Interface*

### 3.2 Feature 2: Project Selection Interface

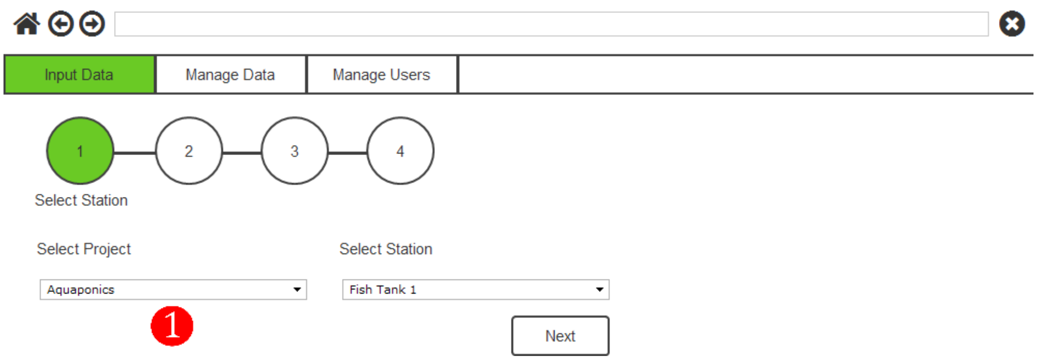


*Figure 6: Project Selection Interface*

Users will select individual projects and stations that require manual input of data. Once selected they can proceed to the input this data in a form which can be accessed from the user’s webpage.

## 3.3 Feature 3: Date Input Interface

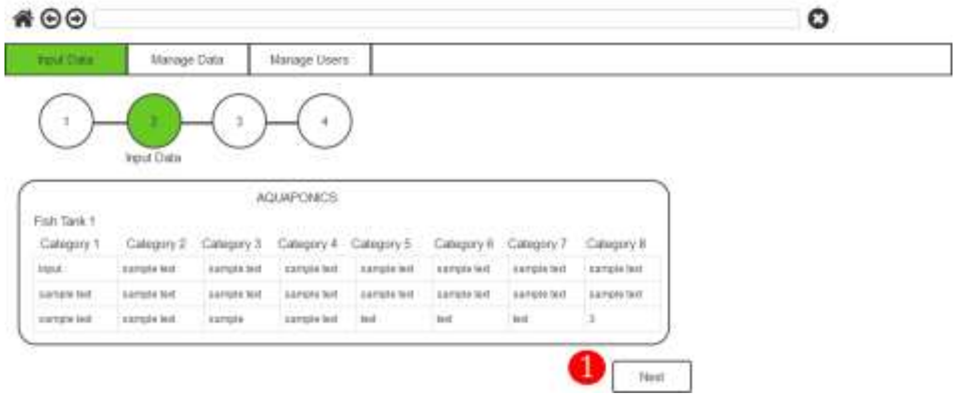


*Figure 7: Data Input Interface*

## 3.4 Feature 4: Review and Submit Data Interface


*Figure 8: Review and Submit Data Interface*

## 3.5 Feature 5: Add/Remove/Edit Users Interface


*Figure 9: Add/Remove/Edit Users Interface*

## 3.6 Feature 6: View/Select Pending Data

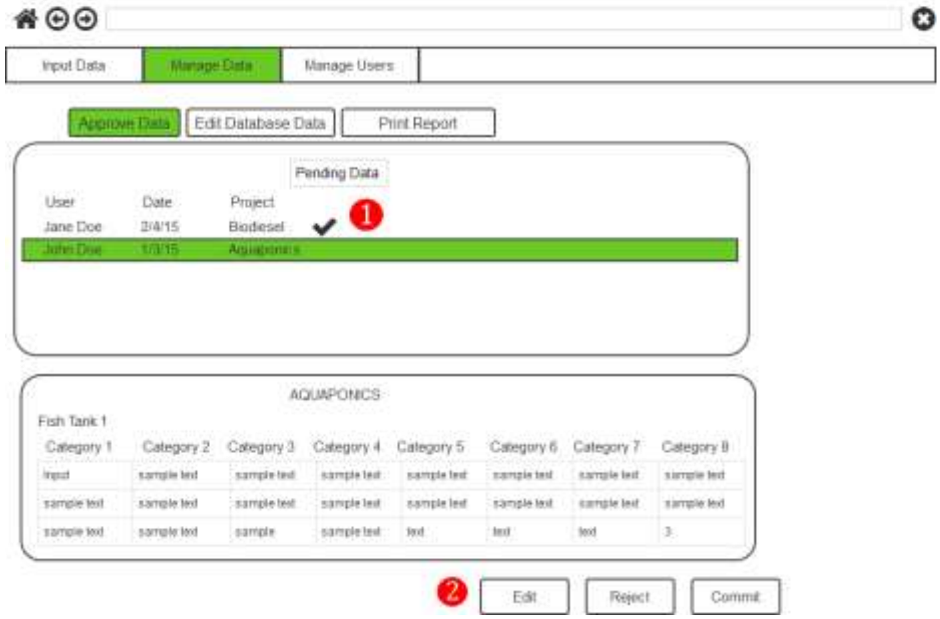


*Figure 10: View/Select Pending Data*

## 3.7 Feature 7: Select/Edit Project of Station Interface


*Figure 11: Select/Edit Project or Station Interface*

# 4. DATABASE SCHEMA

## 4.1 ERD diagram

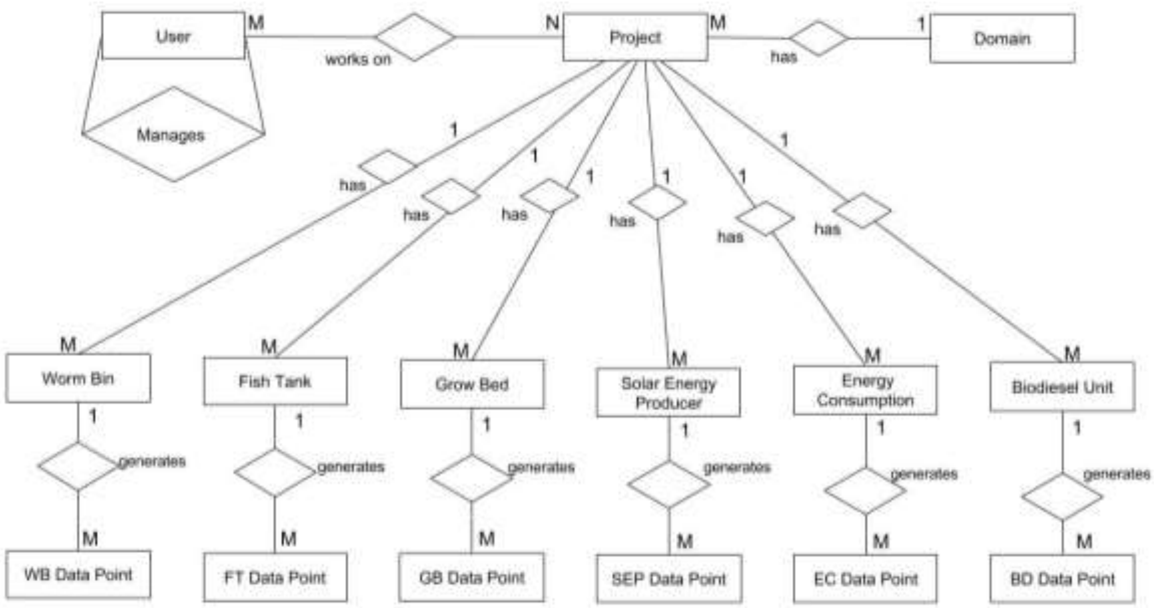


*Figure 12: ERD Diagram*

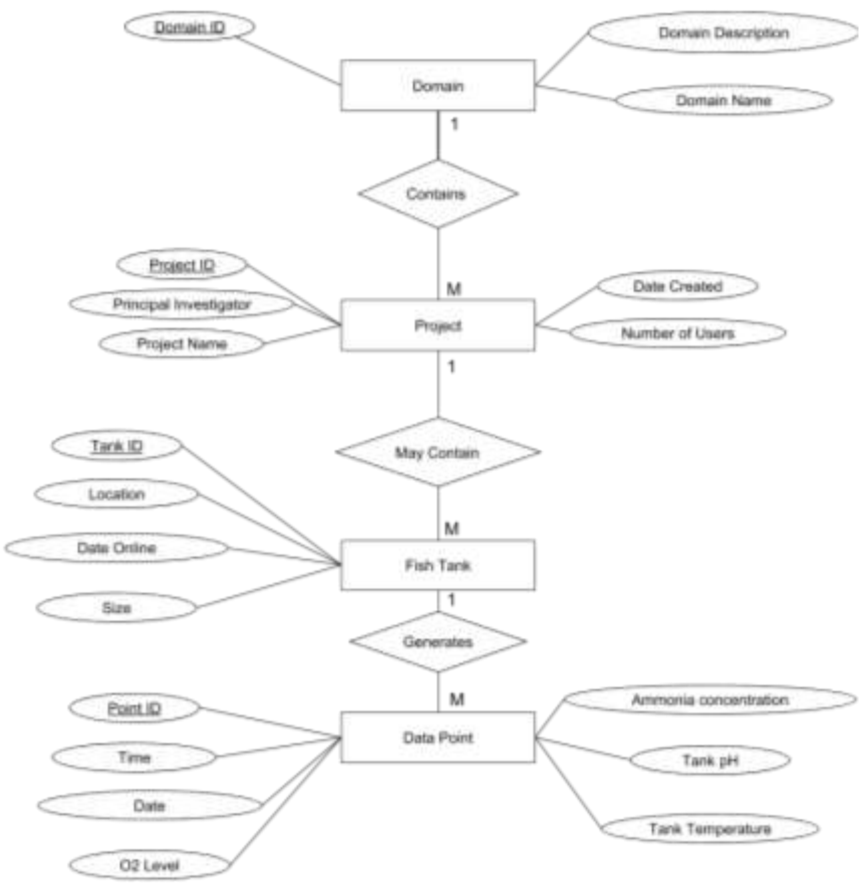


*Figure 13: Project: Fish Tank - Data Point*

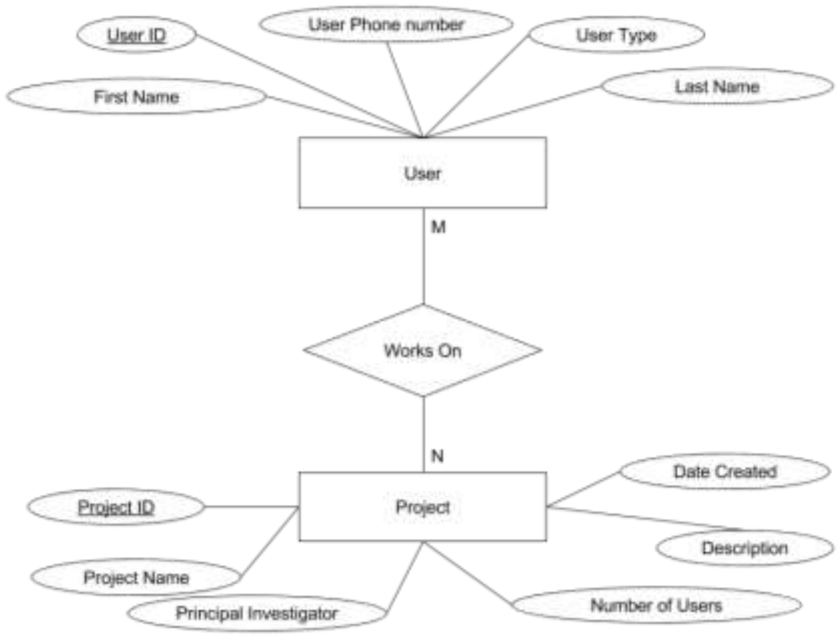


*Figure 14: User - Project*

## 4.2 Creating the database

```
CREATE TABLE Domains (
        DomainID   INT  PRIMARY KEY NOT NULL,
        DomainID   INT  NOT NULL,
        Domain_Name   VARCHAR (50) NULL,
        Domain_Description   VARCHAR (max) NULL
)

CREATE TABLE Projects (
        ProjectID   INT  PRIMARY KEY NOT NULL,
        DomainID   INT  NOT NULL FOREIGN KEY REFERENCES Domains(DomainID),
        Project_Name   VARCHAR (50) NULL,
        Project_description   VARCHAR (max) NULL,
        Principal_Investigator   VARCHAR (50) NULL,
        DATE_Created   DATE  NULL,
        number_of_users   INT  NULL
)

CREATE TABLE Access List (
        ProjectID  INT NOT NULL FOREIGN KEY REFERENCES Projects(ProjectID),
        UserID   INT  NOT NULL FOREIGN KEY REFERENCES Users(UserID),
        Permissions   VARCHAR (50) NULL,
        CONSTRAINT pk_PersonID PRIMARY KEY (ProjectID,UserID)
)
```

```
CREATE TABLE Biodiesel_Stations (
	biodieselID   INT  PRIMARY KEY NOT NULL,
	ProjectID   INT  NOT NULL FOREIGN KEY REFERENCES Project(ProjectID),
	Location   VARCHAR (50) NULL,
	Last_Maintenance_Date   VARCHAR (max) NULL
)

CREATE TABLE Fishtank_Stations (
	FishtankID INT  PRIMARY KEY NOT NULL,
	ProjectID INT  NOT NULL FOREIGN KEY REFERENCES Project(ProjectID),
	Location VARCHAR (50) NULL,
	Name VARCHAR(50) NULL
	Size FLOAT NULL,
	Type_of_Fish  VARCHAR(5) NULL,
	Last_Maintenance_Date   VARCHAR (max) NULL
)

CREATE TABLE Power_Meter_Stations (
	MeterID INT  PRIMARY KEY NOT NULL,
	ProjectID INT  NOT NULL FOREIGN KEY REFERENCES Project(ProjectID),
	Source VARCHAR(50),
	Last_Maintenance_Date   VARCHAR (max) NULL
)


CREATE TABLE Growbed_Stations (
	GrowbedID INT  PRIMARY KEY NOT NULL,
	ProjectID INT  NOT NULL FOREIGN KEY REFERENCES Project(ProjectID),
	Location VARCHAR (50) NULL,
	Name VARCHAR(50) NULL
	SoilType VARCHAR(50) NULL,
	Types_of_Greens	VARCHAR(5) NULL,
	Last_Maintenance_Date   VARCHAR (max) NULL
)

CREATE TABLE Wormbin_Stations (
	WormbinID INT  PRIMARY KEY NOT NULL,
	ProjectID INT  NOT NULL FOREIGN KEY REFERENCES Project(ProjectID),
	Location VARCHAR (50) NULL,
	Name VARCHAR(50) NULL
	Initial_Number_of_Worms INT NULL,
	Volume_Empty		FLOAT NULL,
	Last_Maintenance_Date   VARCHAR (max) NULL
)
```

```
CREATE TABLE Solar_Stations (
	SolarID INT  PRIMARY KEY NOT NULL,
	ProjectID INT  NOT NULL FOREIGN KEY REFERENCES Project(ProjectID),
	Location VARCHAR (50) NULL,
	Panel_Type VARCHAR(50) NULL,
	Energy_Cells INT NULL,
	Number_of_Panels INT NULL,
	Last_Maintenance_Date   VARCHAR (max) NULL
)

CREATE TABLE Users(
	UserID INT  PRIMARY KEY NOT NULL,
	First_Name VARCHAR(50) NULL,
	Last_Name VARCHAR(50) NULL,
	Phone_Number INT NULL
	User_Type VARCHAR(50) NULL,
	Profile_Settings BLOB NULL
)

CREATE TABLE Biodiesel_DataPoint (
	dataID INT  PRIMARY KEY NOT NULL,
	stationID INT  NOT NULL FOREIGN KEY REFERENCES
	Biodiesel_Stations(BiodieselID),
	Methanol_Used FLOAT NULL,
	Potassium_Hydroxide_Used FLOAT NULL,
	Batch_Finish_date DATE NULL,
	Batch_Number INT NULL,
	Batch_Run INT NULL,
	Conversion_Efficiency INT NULL,
	Cost_Kitchen_Oil FLOAT NULL,
	Cost_Waste_Disposal FLOAT NULL,
	Energy_in FLOAT NULL,
	Final_Quality INT NULL,
	Man_Hours INT NULL,
	Market_Price FLOAT NULL,
	Processing_Time TIME NULL
	)

CREATE TABLE Energy_Consumption_DataPoint (
	dataID INT  PRIMARY KEY NOT NULL,
	stationID INT  NOT NULL FOREIGN KEY REFERENCES
	Power_Meter_Stations(MeterID),
	Cost_per_KWHr FLOAT NULL,
	Date DATE NULL,
	Kwhr FLOAT NULL,
	Time TIME NULL
```

```
)

CREATE TABLE Fishtank_DataPoint (
	dataID INT  PRIMARY KEY NOT NULL,
	stationID INT  NOT NULL FOREIGN KEY REFERENCES
	Fishtank_Stations(FishtankID),
	Number_of_Fish INT NULL,
	Ammonia_Concentration FLOAT NULL,
	O2_Concentration FLOAT NULL,
	Temp FLOAT NULL,
	Time TIME NULL,
	Date DATE NULL
)

CREATE TABLE Growbed_DataPoint (
	dataID INT  PRIMARY KEY NOT NULL,
	stationID INT  NOT NULL FOREIGN KEY REFERENCES
	Growbed_Stations(GrowbedID),
	Humidity FLOAT NULL,
	Temp FLOAT NULL,
	Number_of_Greens INT NULL,
	Production_Rate FLOAT NULL,
	Time TIME NULL,
	Date DATE NULL
)

CREATE TABLE Solar_DataPoint (
	dataID INT  PRIMARY KEY NOT NULL,
	stationID INT  NOT NULL FOREIGN KEY REFERENCES Solar_Stations(SolarID),
	KWHr FLOAT NULL,
	Light_Meter_Reading FLOAT NULL,
	Temp FLOAT NULL,
	Time TIME NULL,
	Date DATE NULL
)

CREATE TABLE Wormbin_DataPoint (
	dataID INT  PRIMARY KEY NOT NULL,
	stationID INT  NOT NULL FOREIGN KEY REFERENCES
	Wormbin_Stations(WormbinID),
	Bed_Moisture_Concentration FLOAT NULL,
	Volume_Produced FLOAT NULL,
	Temp FLOAT NULL,
	Time TIME NULL,
	Date DATE NULL
)
```

# 5. COMPONENT DESIGN SPECIFICATIONS

| | **Use Case Components** | | **Database** |
|---|---|---|---|
| Feature | Webpage | Use Case | Table/Relations |
| Manage Data | ManageData.html<br>ManageData.cs<br>ManageData.js<br>ManageData.aspx | UC1. Manage Data | |
| Manage Users | ManageUsers.html<br>ManageUsers.cs<br>ManageUsers.js<br>ManageUsers.aspx | UC2. Manage Users | |
| Input Data | InputData.html<br>InputData.cs<br>InputData.js<br>InputData.aspx | UC3. Input Data | |
| Customize Dashboard | ManageWid.html<br>ManageWid.cs<br>ManageWid.js<br>ManageWid.aspx<br>AddWid.html<br>AddWid.cs<br>AddWid.js<br>AddWid.aspx<br>EditWid.html<br>EditWid.cs<br>EditWid.js<br>EditWid.aspx<br>DeleteWid.html<br>DeleteWid.cs<br>DeleteWid.js<br>DeleteWid.aspx | UC4. Customize Dashboard | |
| Login | Login.html<br>Login.js | UC5. Login | |

CD Jam will be using the coding standards stated in *APPENDIX C: Coding Standards* in order to code the STORC Dashboard Project. Many of these standards have been compiled from individual group knowledge obtained while working for internships, knowledge from classes at CSUS, and rules the Sponsor as well as CD Jam deem important for code maintainability.

## 5.1 Sequence Diagrams (Introduction)

### 5.1.1 Component 1: Add User Sequence Diagram

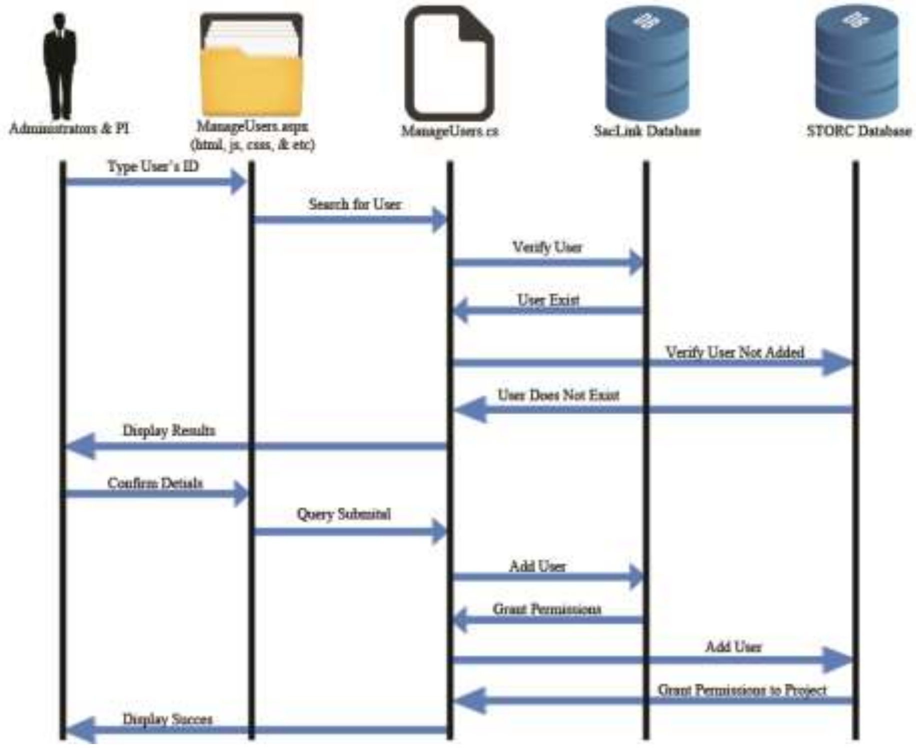


*Figure 15: Add User*

### 5.1.2 Component 2: Change User Permissions Sequence Diagram

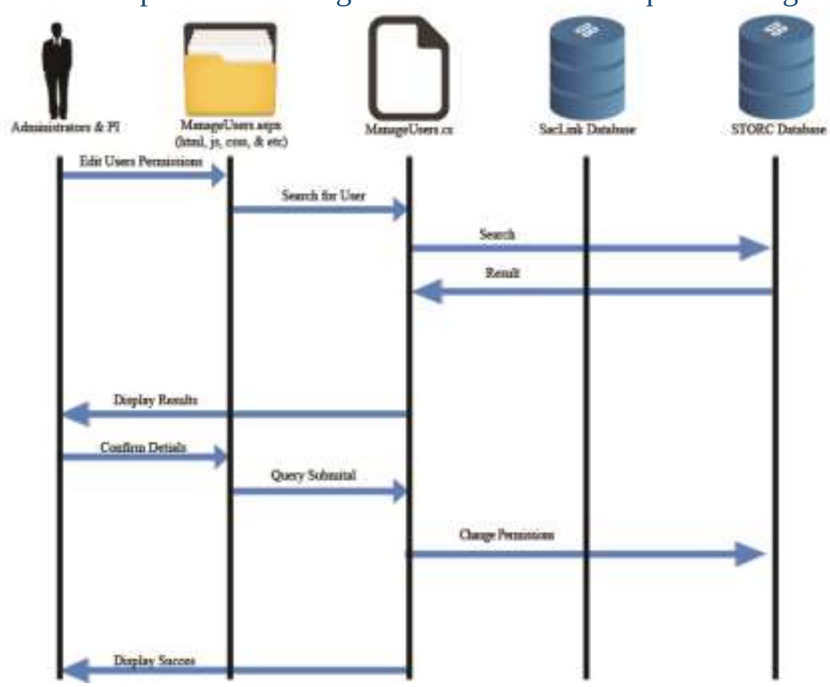


*Figure 16: Change User Permissions*

### 5.1.3 Component 3: Data Input Sequence Diagram

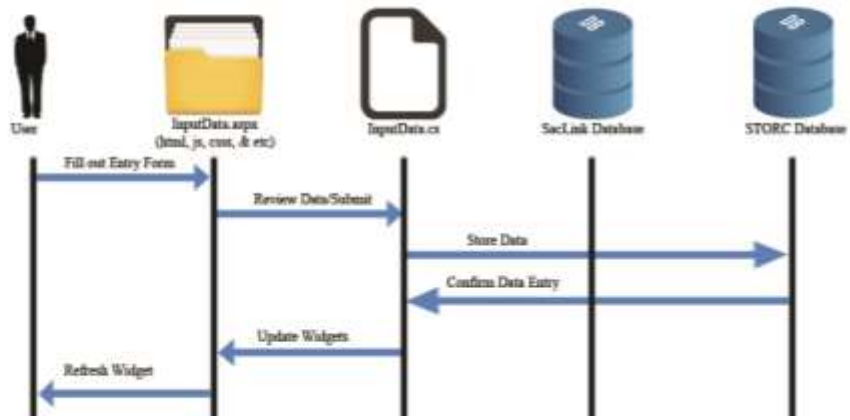


*Figure 17: Data Input*

### 5.1.4 Component 4: Edit Data Sequence Diagram

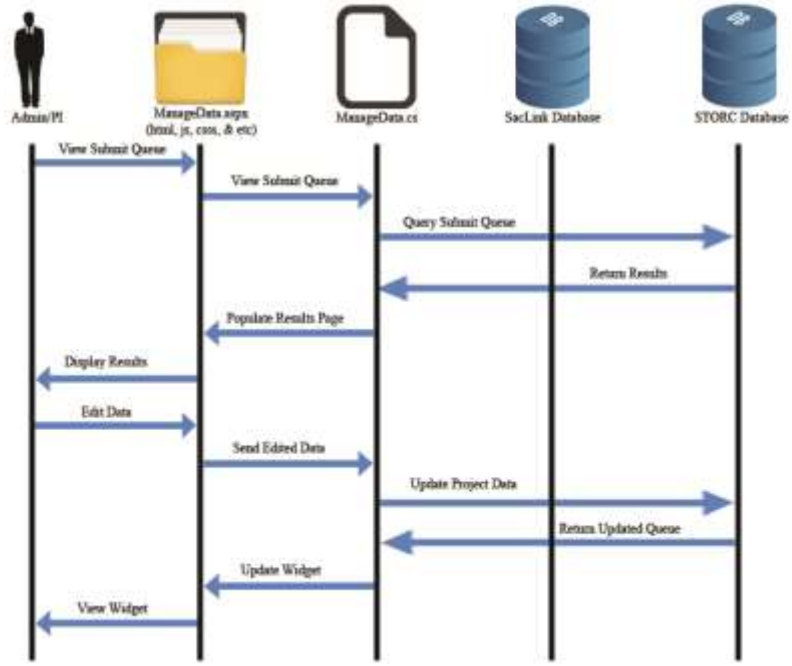


*Figure 18:Edit Data*

### 5.1.5 Component 5: Remove Data Sequence Diagram

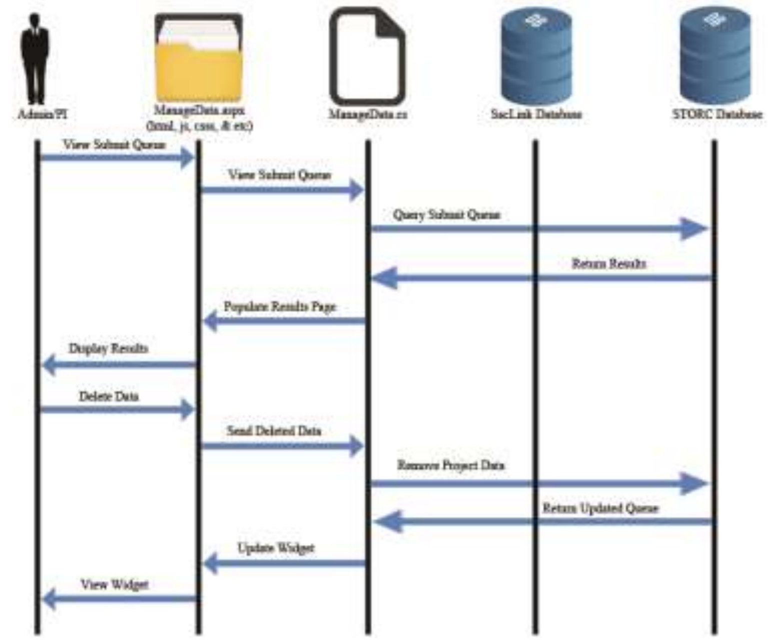


*Figure 19: Remove Data*

### 5.1.6 Component 6: Login Sequence Diagram

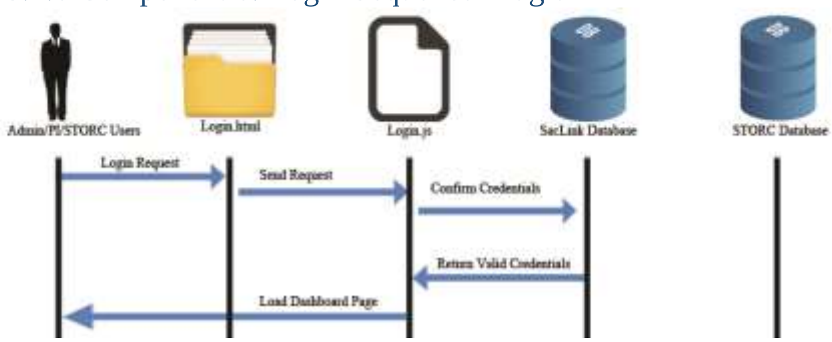


*Figure 20: Login*

### 5.1.7 Component 7: Report Data Sequence Diagram


*Figure 21: Report Data*

### 5.1.8 Component 8: Remove User Sequence Diagram

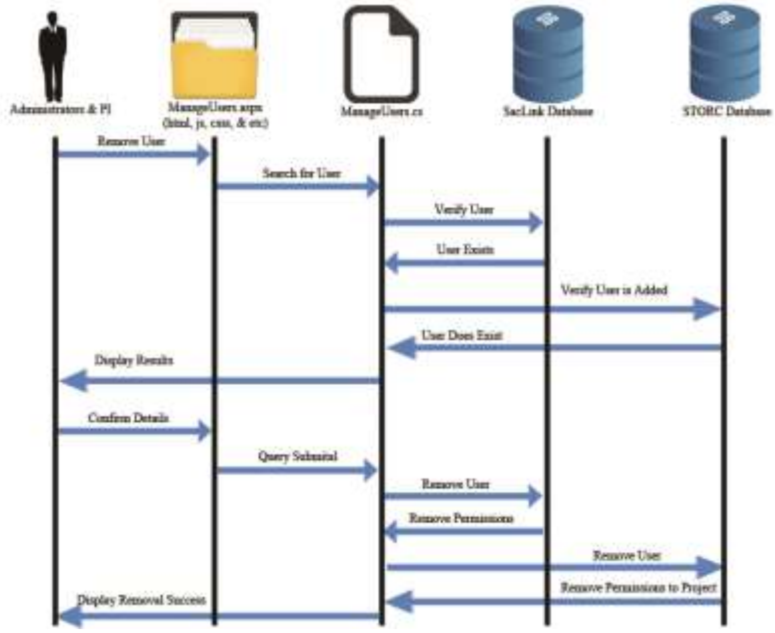


*Figure 22: Remove User*

### 5.1.9 Component 9: Create New Project Sequence Diagram

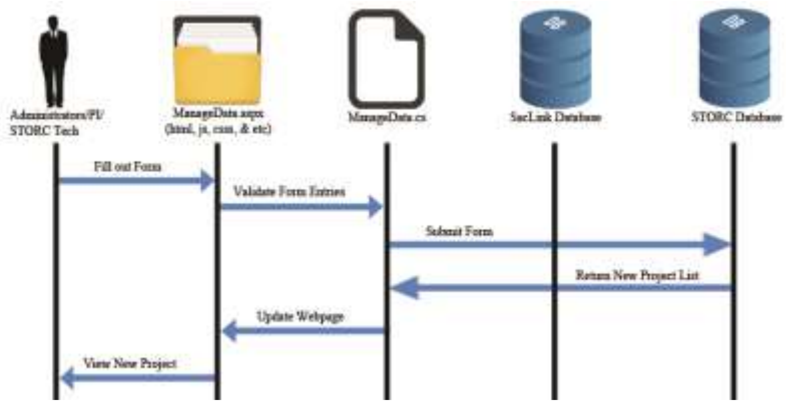


*Figure 23: Create New Project*

### 5.1.10 Component 10: Add Widget Sequence Diagram

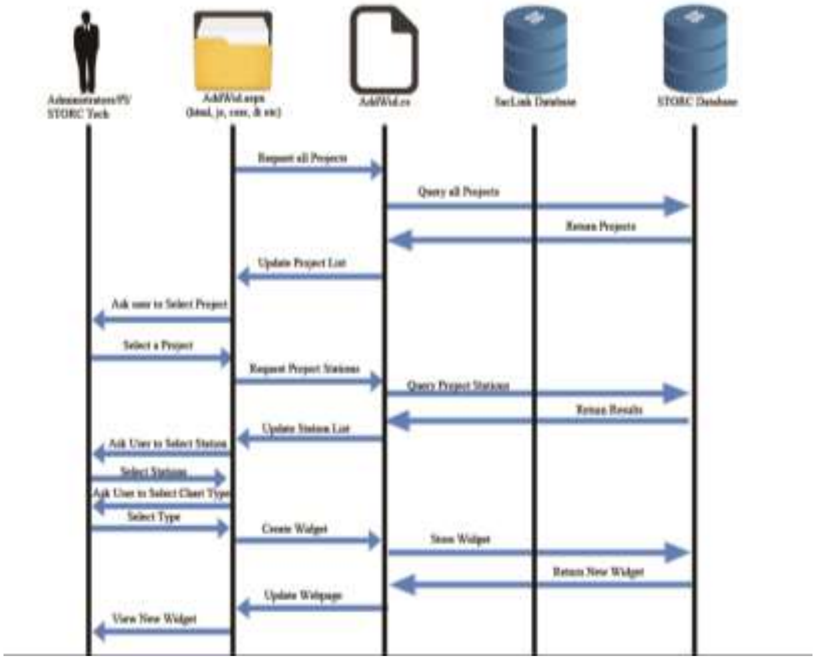


*Figure 24: Add Widget*

### 5.1.11 Component 11: Remove Widget Sequence Diagram

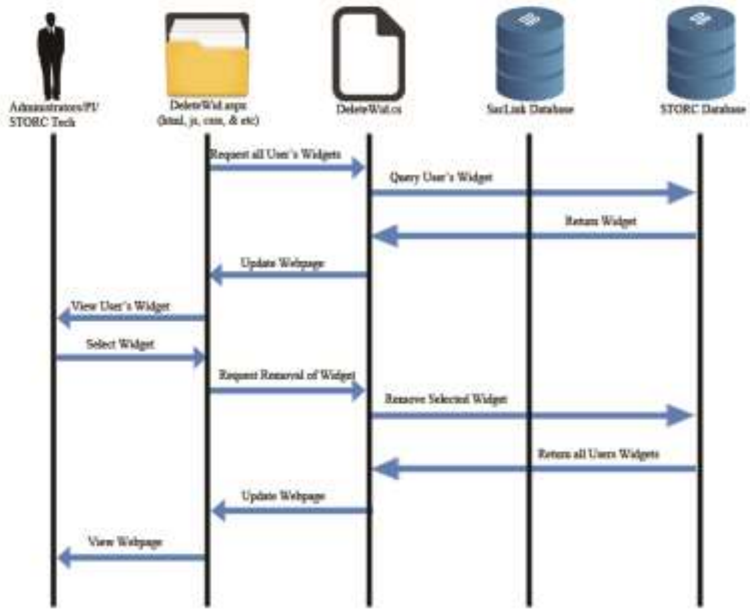


*Figure 25: Remove Widget*

### 5.1.12 Edit Widget

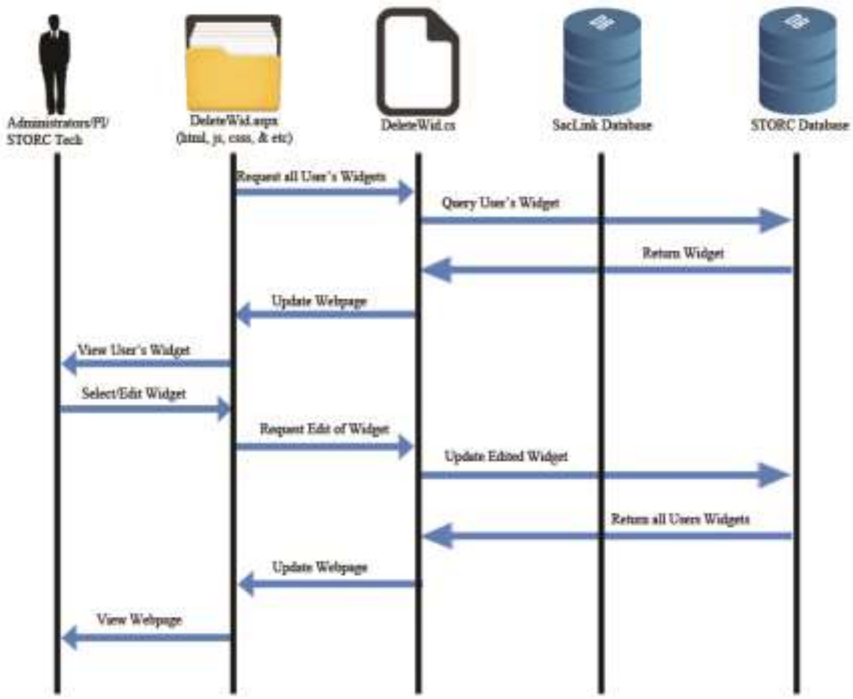


*Figure 26: Edit Widget*

## 5.2 Webpage and Function Design Specifications

### 5.2.1 Dashboard.apx

Preconditions: User navigates to the STORC webpage
Interface: STORC Dashboard website default homepage
Processing Specifications: Load all default widgets on homepage
Screen Print: Prints all widgets and homepage to the screen
Database Requirements: None
Postconditions: User leaves the webpage

### 5.2.2 Dashboard.cs

Preconditions: User navigates to the STORC webpage
Interface: STORC Dashboard website default homepage
Processing Specifications: Handles backend of homepage management
Screen Print: None
Database Requirements: Query database for data required by the widgets
Postconditions: None

### 5.2.3 ManageWid.aspx

Preconditions: User navigates to the STORC web page and user must be logged in
Interface: Widget Management Interface
Processing Specifications: Request all user's widgets
Screen Print: Show all widgets for the user
Database Requirements: Handled by ManageWid.cs
Postconditions: User leaves webpage

### 5.2.4 ManageWid.cs

Preconditions: User must access the application
Interface: The component doesn't provide interface

Processing specifications:  Handles backend widget management
Screen print: None
Database requirements: The user must have a widget
Postconditions: Widget is updated

#### 5.2.5 ManageUsers.aspx

Preconditions: User must access application and be authorized to manage users
Interface:  Manage User Interface
Processing Specifications:  None
Screen Print: Display all users that are manageable
Database Requirements:   Managed by ManageUsers.aspx
Postconditions: Dashboard Homepage is displayed

#### 5.2.6 ManageUsers.cs

Preconditions: User accessed  application and is authorized to manage users
Interface:  The component doesn't provide an interface
Processing Specifications: The component handles backend user management
Screen Print: None
Database Requirements:   Access to User tables
Postconditions: ManageUsers.aspx is updated

#### 5.2.7 InputData.aspx

Preconditions: User must access application and be authorized to input project data
Interface:  Data Input Webform
Processing Specifications: Request data to be stored in Input Data Queue
Screen Print: Show confirmation page
Database Requirements: None
Postconditions: Refresh Dashboard Homepage

#### 5.2.8 InputData.cs

Preconditions: User accesses application and is authorized to input project data
Interface:  This component doesn't provide an interface
Processing Specifications: Add data to the Input Queue
Screen Print: None
Database Requirements: Access to Input Queue
Postconditions: Refresh InputData.aspx

#### 5.2.9 ManageData.aspx

Preconditions: User accesses application and is authorized to manage project data
Interface:  Data Management Interface
Processing Specifications: None
Screen Print: Displays a revised data table
Database Requirements: Handled by ManageData.cs
Postconditions: User returns to Dashboard Home

#### 5.2.10 ManageData.cs

Preconditions: User accesses application and is authorized to manage project data
Interface: This component doesn't provide an interface
Processing Specifications: Queries project tables for data management
Screen Print: None
Database Requirements: Access to specified project tables
Postconditions: Updates ManageData.aspx

#### 5.2.11 Login.html

Preconditions: Users not logged in, connected to a network
Interface: Login Screen
Processing Specifications: Queries Saclink Database
Screen Print: Confirmation Page and/or Error Message
Database Requirements: Saclink Database must be running
Postconditions: None

# 6. PERFORMANCE ANALYSIS

The most relevant performance constraints affecting our project relates to speed. Each widget will receive data from the server and create graphs based on the user's preferences. Performing these computations and creating graphs takes time and resources. In order to reduce the amount of time and increase throughput, CD Jam will make efforts to reduce the traffic on the STORC servers. This is done by using a server to perform heavy operations such as widget graphic generation, data validation, and look-and-feel configuration. The client side portion of the STORC Dashboard project will focus on the look and feel of the web pages and widgets, thus avoiding unnecessary server CPU and RAM load.

Due to budget constraints STORC will no longer use sensors to monitor projects. Without sensors, all data must be entered manually by students or staff members. This makes it impossible to show real-time data.

# 7. RESOURCE ESTIMATES

IRT will host the STORC Dashboard Project. The web content and database should require no more 500GB of storage space. If more space in needed later, incoming Senior Project classes can expand the size of the database. In the future STORC may add in sensors and data collection will become more of an automatic process.

# 8. SOFTWARE REQUIREMENTS TRACEABLITY MATRIX

| Associations (Sections) | | |
|---|---|---|
| **Feature** | **SDS** | **SRS** |
| Manage Data | 3.1 Homepage Management Interface<br>3.4 Review and Submit Data Interface | 2.3.4 UC-3 Manage Data |
| Manage Users | 3.5 Add/Remove/Edit Users Interface | 2.3.3 UC-2 Manage Users |
| Input Data | 3.3 Data Input Interface,<br>3.6 View/Select Pending Data | 2.3.2 UC-3 Input Data |
| Customize Dashboard | 3.2 Project Selection Interface<br>3.7 Select/Edit Project or Station Interface | 2.3.1 UC-4 Customize Dashboard |
| Login | Not Shown Configured by IRT | Not Shown Configured by IRT |

# 9. APPROVALS

By signing you agree that all conditions and commitments to the project are accurate to the best of your knowledge. I certify that the information in this Software Project Management Plan is correct and the senior project group *CD Jam* can continue on with the design of the project. I also certify that I will follow and provide all needing requirements stated in this document and that I am willing to follow through with all conditions.

**CD Jam Team members:**

X ______________________________
Cole Culler

X ______________________________
David Grapentine
Project Lead

X ______________________________
Ashley Gregory

X ______________________________
John Jones

X ______________________________
Michael Smith

**Faculty Advisor:**

X ______________________________
Ying Jin
Faculty Advisor

# APPENDIX A: Database tables and attributes

## Access List Table

| Element | Description | How it is Set | How it is Used | Value Type | List of Discrete Values | Range |
|---|---|---|---|---|---|---|
| Permissions | Limits access of users | User input | Sets user permissions in the application | Integer | Must be 9 digits long | 0-9 |
| Project ID | A project's unique ID | Automatically by database | identifies specific projects | Integer | Must be 9 digits long | 0-9 |
| User ID | A user's unique ID | Acquired from CSUS ID | identifies specific users | Integer | Must be 9 digits long | 0-9 |

## Biodiesel Station Table

| Element | Description | How it is Set | How it is Used | Value Type | List of Discrete Values | Range |
|---|---|---|---|---|---|---|
| Date Online | Date a station came online | Manually entered by database manager | Monitors a station's uptime | Date | N/A | N/A |
| Last Maintenance Date | The last data the station was maintained | Manually entered by the user | Monitors maintenance schedule | Date | N/A | N/A |
| Location | Location of the Biodiesel Station | Manually entered by database manager | Stores the location of the project | varchar(30) | N/A | A-Z, a-z |
| Project ID | A project's unique identification | Created by database | Used to identify specific projects | Integer | N/A | N/A |
| Station ID | A station's unique identification | Created by database | Used to identify specific stations | Integer | N/A | N/A |

### Biodiesel DataPoint Table

| Element | Description | How it is Set | How it is Used | Value Type | List of Discrete Values | Range |
|---|---|---|---|---|---|---|
| Amount of Methanol Used | Contains the amount of methanol used | Manually by user | Used as a factor to determine the sustainability of a project | Float | N/A | N/A |
| Amount of Potassium Hydroxide Used | Contains the amount of Potassium Hydroxide used | Manually by user | Used as a factor to determine the sustainability of a project | Float | N/A | N/A |
| Batch Finish Date | Lists the date when a batch of biodiesel finished production | Manually by user | Used to show date biodiesel batch finished production | Date | N/A | N/A |
| Batch Number | Used to identify the batch number of biodiesel produced | Manually by user | Keeps track of the biodiesel produced by a biodiesel station | Integer | N/A | N/A |
| Batch Run | Used to identify the batch run of biodiesel produced | Manually by user | Keeps track of the biodiesel produced by a biodiesel station | Integer | N/A | N/A |
| Conversion Efficiency | Shows the energy used to produced the biodiesel per gallon of fuel | Manually by user | Used to show how efficient it is to make biodiesel | Integer | Must be a single digit | 1-10 |
| Cost of Kitchen Waste Oil | Cost to purchase kitchen waste oil | Manually by user | Used as a factor in sustainability calculations | Float | N/A | N/A |
| Cost of Waste Disposal | Cost to dispose of all waste produced in the biodiesel making process | Manually by user | Used as a factor in sustainability calculations | Float | N/A | N/A |
| Datapoint ID | ID of the data point | Createf by database | Used to identify specific projects | Integer | N/A | N/A |
| Energy Required to produce Biodiesel | The overall amount of energy needed to produce biodiesel | Manually by user | Used as a factor in sustainability calculations | Float | N/A | N/A |

| | | | | | | |
|---|---|---|---|---|---|---|
| Final Quality | Grade of the quality of diesel produced | Manually by user | Used as a factor in optimization calculations | Integer | Must be a single digit | 1-10 |
| Man-hours | Man hours spent on a project | Manually by user | Used as a factor in sustainability calculations | Integer | N/A | N/A |
| Market Cost of Biodiesel | The market cost per gallon of biodiesel | Manually by user | Used as a factor in sustainability calculations | Float | N/A | N/A |
| Station ID | A station unique identification | Created by database | Used to identify specific stations | Integer | N/A | N/A |
| Total Processed Time | Total time needed to produced a batch of biodiesel | Entered in by a user manually | Total amount of time spend producing a batch of biodiesel | Time | N/A | N/A |
| Volume Biodiesel Produced | The amount of biodiesel produced, measured in gallons | Entered in by a user manually | The measurement in gallons of the amount of biodiesel produced | Float | N/A | N/A |
| Volume Kitchen Waste Oil Collected | The amount of kitchen oil waste used in production of biodiesel | Entered in by a user manually | The measurement in gallons of the amount of kitchen oil used in the production batch of biodiesel | Float | N/A | N/A |
| Volume Kitchen Waste Oil Produced | The waste of the kitchen oil left after the production of biodiesel | Entered in by a user manually | The measurement in gallons of the amount of kitchen oil waste produced in the production batch of biodiesel | Float | N/A | N/A |
| Volume Waste Produced | The total amount of waste produced in the biodiesel making process | Entered in by a user manually | The amount of waste produced in the overall production of a batch of biodiesel | Float | N/A | N/A |

### Domain Table

| Element | Description | How it is Set | How it is Used | Value Type | List of Discrete Values | Range |
|---|---|---|---|---|---|---|
| Domain Description | A domain is a group of projects. | Manually by PI | Short description explaining the domain | String | N/A | N/A. |
| Domain ID | A unique ID for | Automatically by database | Uniquely identifies the domain. | Int | N/A | N/A |
| Domain Name | This will be the actual name for each domain. | Manually by PI | This will be used to identify each domain. | Varchar | N/A | N/A |

### Energy Consumption DataPoint Table

| Element | Description | How it is Set | How it is Used | Value Type | List of Discrete Values | Range |
|---|---|---|---|---|---|---|
| Domain Description | A domain is a group of projects. | Manually by PI | Short description explaining the domain | String | N/A | N/A. |
| Domain ID | A unique ID for | Automatically by database | Uniquely identifies the domain. | Int | N/A | N/A |
| Domain Name | This will be the actual name for each domain. | Manually by PI | This will be used to identify each domain. | Varchar | N/A | N/A |

### Fishtank_Stations Table

| Element | Description | How it is Set | Value Type | List of Discrete Values | Range |
|---|---|---|---|---|---|
| Initial Number of Fish | Number of fish in the tank when project is set up | User Input | Int | N/A | N/A |
| Location | The location of the tank within the STORC site | User Input | Varchar | N/A | N/A |
| Name | Name of the tank | User Input | Varchar | N/A | N/A |
| Project ID | Number that identifies the project | Database default | int | N/A | N/A |
| Size | The size of the tank in gallons | User input | Float | N/A | N/A |
| Tank ID | Number that identifies the tank | Database default | int | N/A | N/A |
| Type of Fish | The types of fish in the tank | User Input | Varchar | N/A | N/A |

### Fishtank DataPoint Table

| Element | Description | How it is Set | Value Type | List of Discrete Values | Range |
|---|---|---|---|---|---|
| Datapoint ID | Number that identifies the data point | Database default | Int | N/A | N/A |
| Date | The date the data point was captured. | User Input | Date | N/A | 1000-01-01 to 9999-12-31 |
| Number of Fish | The number of fish currently in the tank | User Input | Int | N/A | N/A |
| Tank Ammonia Concentration | Concentration of Ammonia that is in the tank | Sensor Input | Float | N/A | N/A |
| Tank ID | Number that identifies the tank ID | Database Default | Int | N/A | N/A |
| Tank $O_2$ Concentration | Amount of $O_2$ concentration in the tank | Sensor Input | Float | N/A | N/A |
| Tank Temperature | The tank temperature | Sensor Input | Float | N/A | N/A |
| Time | The time the data point was captured | User Input | Time | N/A | -838:59:59 – 838:59:59 |

### Growbed Stations Table

| Element | Description | How is it set | How is it used | Value Type | Discrete Values | Range |
|---|---|---|---|---|---|---|
| Bed ID | Number to distinguish grow beds | user input | Used to distinguish multiple grow beds | int | N/A | N/A |
| Date Online | Date when grow bed project begins | user input | Used to specify range for analytics | Date | N/A | 1000-01-01 to 999-12-31 |
| Location | The STORC location of grow bed project | user input | Maintenance and alerts | Varchar | N/A | N/A |
| Project ID | Number to distinguish projects | automatically | Used to distinguish between multiple projects | Int | N/A | N/A |
| Soil Type | The type of soil used in grow bed | user input | Used in grow bed calculations | Varchar | N/A | N/A |
| Types of Greens | Type of greens produced in grow bed | user input | Used in grow bed calculations | Varchar | N/A | N/A |

### Growbed DataPoint Table

| Element | Description | How is it set | How is it used | Value Type | Discrete values | Range |
|---|---|---|---|---|---|---|
| Bed ID | Number to distinguish grow beds | user input | Used to distinguish multiple grow beds | Int | N/A | N/A |
| Datapoint ID | Number to distinguish data entries | automatically | Used to distinguish multiple data entries | Int | N/A | N/A |
| Date | Date that the data point is recorded | automatically | used to specify date range for analytics | Date | N/A | 1000-01-01 to 999-12-31 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Greenhouse Humidity | Measurement of Humidity at grow bed location | user input | Used as a factor in growth rate calculations | Float | N/A | N/A |
| Greenhouse Temperature | Measurement of Temperature at grow bed location | user input | Used as a factor in growth rate calculations | Float | N/A | N/A |
| Number of Greens | Number of greens units produced | user input | Used as a factor in growth rate calculations | Int | N/A | N/A |
| Production Rate | The rate at which greens are produced | user input | referenced for grow bed analytics | Int | N/A | N/A |
| Time | Time data point is recorded | automatically | Used to specify time grow bed data point is recorded | Time | N/A | -838:59:59 To 838:59:59 |

### Projects Table

| Element | Description | How it is Set | How it is Used | Value Type | List of Discrete Values | Range |
|---|---|---|---|---|---|---|
| Project ID | Unique ID for the Project. | Automatically by database | Uniquely identifies project | Int | N/A | N/A |
| Domain ID | Unique ID for the domain | Automatically by database | Uniquely identifies domain | Int | N/A | N/A |
| Project Name | Name of project | Manually by PI | This is used to quickly find a particular project from the table. | Varchar | N/A | N/A |
| Project Description | Short description of project | Manually by PI | Short description of project | Varchar | N/A | N/A |
| Principal Investigator | Each project will have a PI. This information must go in the project table. | The webmaster will enter the PI information. | identifies project PI | Varchar | N/A | N/A |
| Date Created | This will describe when a particular project is created. | This will be manually entered by a database manager | Keeps track of particular project was created | Int | N/A | N/A |

| | | | | | | |
|---|---|---|---|---|---|---|
| Number of Users | This will describe how many users are assigned to a certain project. | This will automatically be generated each time a user is added. | Keeps track of number of users per project | Int | N/A | N/A |

### Solar Stations table

| Element | Description | How it is Set | How it is Used | Value Type | List of Discrete Values | Range |
|---|---|---|---|---|---|---|
| Date Online | This information displays the date that each solar panel goes online | This data will be entered manually by PI or webmaster | This data will help users know when a certain solar panel went online | Int | N/A | N/A |
| Energy Cells | This data will keep track of all of the energy cells involved with thermal heat transfer | This data will be entered manually by either the PI or database manager. | This data is used solely to keep track of the energy cells used for each project | Int | N/A | N/A |
| Location | Location data pertains to the location of each solar unit | Manually entered by PI | Used to keep track of station locations | Varchar | N/A | N/A |
| Number of Panels | This data portrays the amount of panels at STORC | PI’s will enter this data manually | This data is used to show people at STORC the number of solar panels in use | Int | N/A | N/A |
| Panel Type | This will show users what type of solar panels being used for a particular project | PI’s or database manager will manually create this data | This data is used to inform the user of the type of panel being used | Varchar | N/A | N/A |
| Project ID | This data is used to identify a particular project | Automatically by database | This data is used to organize and identify the various projects related to solar heat transfer | Int | N/A | N/A |
| Station ID | Unique ID for Solar Thermal Station | Automatically by database | Uniquely identifies station | Int | N/A | N/A |

### Solar DataPoint Table

| Element | Description | How is it set | How is it used | Value Type | Discrete values | Range |
|---|---|---|---|---|---|---|
| Ambient Temperature | The local ambient atmospheric temperature | User input | To be able to do calculations that my require the outside temperature | Float | N/A | N/A |
| Datapoint ID | Number to distinguish solar power data entries | Automatically | Used to distinguish multiple solar power data entries | Int | N/A | N/A |
| Date | Date that the data point is recorded | Automatically | Used to specify date range for analytics | Date | N/A | 1000-01-01 to 999-12-31 |
| Kilowatts per Hour | Amount of energy consumed by STORC | User input | Used to track STORC sustainability. Contrasted against STORC energy produced | Float | N/A | N/A |
| Light Meter Reading | Measurement of amount of light hitting solar panel | User input | Used whenever widget references the value | Float | N/A | N/A |
| Station ID | Number to distinguish different stations | Automatically | Used to distinguish multiple stations | Int | N/A | N/A |
| Time | Time data point is recorded | Automatically | Used to specify time solar power data point is recorded | Time | N/A | -838:59:59 To 838:59:59 |

### Wormbin Stations Table

| Element | Description | How it is Set | How it is Used | Value Type | List of Discrete Values | Range |
|---|---|---|---|---|---|---|
| Bin ID | This is used to identify the different worm bins on site. | Automatically by database | Bin ID is used to identify each individual bin at STORC. | Int | N/A | N/A |
| Initial Number of Worms | This number represents the initial number of worms for a particular bin. | This will be manually entered by the database manager | This is used solely to keep track of the original number of worms in a bin. | Int | N/A | N/A |
| Location | This will describe where each bin is located at STORC. | This information will be entered manually by the database manager | This will be used periodically to notify users where each bin is | Varchar | N/A | N/A |
| Name | This will identify a name for each individual worm bin | This information is setup manually by the database manager | The bin name is used to identify each individual bin at STORC | Varchar | N/A | N/A |
| Project ID | This data is used to identify each worm bin project | Automatically by database | This set of data is used to identify each project associated with the worm bins | Int | N/A | N/A |

| | | | | | | |
|---|---|---|---|---|---|---|
| Size | This data specifies the size of each worm bin | This is manually entered by either the webmaster or database manager | This data is used to inform users how many gallons large each worm bin is | Float | N/A | N/A |
| Volume Empty | This data shows the volume of the bin | This is manually entered when creating the project | This data is used to show the initial volume of the worm bin when they are empty | Float | N/A | N/A |

## Wormbin DataPoint Table

| Element | Description | How is it set | How is it used | Value Type | Discrete Values | Range |
|---|---|---|---|---|---|---|
| Bed Moisture Concentration | Measurement of worm bin moisture | User input | Used in vermicomposting calculations | Float | N/A | N/A |
| Bed Temperature | Measurement of worm bin temperature | User input | Used in vermicomposting calculations | Float | N/A | N/A |
| Bin ID | Number to distinguish worm bins | Automatically by database | Used to distinguish multiple worm bins | Int | N/A | N/A |
| Datapoint ID | Number to distinguish data entries | Automatically | Distinguishes multiple data entries | Int | N/A | N/A |
| Date | Date that the data point is recorded | Automatically | used to specify date range for analytics | Date | N/A | 1000-01-01 to 999-12-31 |
| Time | Time data point is recorded | Automatically | Used to specify time grow bed data point is recorded | Time | N/A | -838:59:59 To 838:59:59 |
| Volume Produced | The volume of vermicomposting produced | user input | Used in vermicomposting calculations | Float | N/A | N/A |

## Users Table

| Element | Description | How is it set | How is it used | Value Type | Discrete Values | Range |
|---|---|---|---|---|---|---|
| First Name | The first name of a student. | Manually by PI | This data is used to identify a certain user. | Varchar | N/A | N/A |
| Last Name | The last name of a student. | Manually by PI | This data is used to identify a certain user. | Varchar | N/A | N/A |
| Phone Number | The phone number entered is associated with the individual student. | Manually by PI | This data will also be used to identify and contact users. | Varchar | N/A | N/A . |
| Profile Settings | This entails the organization and types of widgets in the dashboard. | Any user with access to the dashboard will be able to save their settings. | This will be used to save the way the dashboard looks for the user. | BLOB | N/A | N/A |
| User ID | Sacramento State Identification number will be attached to each student. | Manually by PI | This is another form of identification for the user. | Int | N/A | N/A |
| User Type | This will describe the type of access the user will have in the dashboard. | Manually by PI or Admin | This is used to identify the type of access the user has in the dashboard. | Varchar | N/A | N/A |

# APPENDIX B: Alphabetic listing of each attribute and its characteristics

| Table Name | Descriptions | How it is Set | How it is Used | Expected Growth of Records | Record Size in Bytes |
|---|---|---|---|---|---|
| Access List | Links a user to a STORC Project and sets their permissions | Set manually by an administrator | The permissions coupled with the User ID and Project ID limits each user to their list of projects | The records a will grow and shrink each semester. There will be at most 60 people | 12 bytes |
| Biodiesel Data Point | The daily activities that STORC employees undergo are recorded here | Manually by STORC employees | This data is closely monitored by STORC users in order to measure the level of sustainability | The record of data points will constantly be growing as the number of projects increase | 82 bytes |
| Biodiesel Station | Links users to a specific biodiesel station and project | Either the PI or webmaster will manually enter this data | This data is used to help STORC employees keep track of the different biodiesel stations | The record of data will grow as biodiesel projects are added and decrease as projects finish their course | 54 bytes |
| Domain | Consists of all project types/domains | Entered manually by webmaster | Links specific projects to appropriate project domain | Expected rate of growth should be low | 8034 bytes |
| Energy Consumption Data | This data describes how much energy STORC is pulling from the grid of Sacramento | This data will be automatically entered. After the servers record the information, the back end program will | Energy consumption data is used to give STORC employees an idea of their level of sustainability | Kilowatts per hour and cost per kilowatt per hour will fluctuate depending on STORC's activity. | 38 bytes |

| | | | | | |
|---|---|---|---|---|---|
| | | calculate energy consumption | | | |
| Energy Consumption Data Point | The daily activities at STORC that relate to energy consumption are manually recorded | Any STORC user with correct access will be able to manually enter relevant data | This data is very important for monitoring the level of sustainability at STORC | The number of records for energy consumption will constantly grow | 30 bytes |
| Fish Tank Data Point | All daily activities related to Fish tank stations will have data that needs to be entered in constantly | STORC employees will be manually entering in this data as part of their daily duties | The data points collected will be used to monitor the overall efficiency and functionality of these stations | This number of records will grow quickly, but overtime will be compressed to make room for new data | 38 bytes |
| Fish Tank | Links users to specific Fish Tank ID’s | The PI or webmaster will manually enter this data | This is used to help STORC employees keep track of the different Fish tanks and the data that correlates with each one | This number of records will grow quickly, but overtime will be compressed to make room for new data | 80 bytes |
| Grow Bed | This table will contain all data related to the grow bed stations | Some of this data, like soil temperature will be recorded automatically. Other data will have to be recorded manually. | This data will be used to monitor the development and health of the grow beds. | Depending on the number of grow beds active, the amount of data may either increase or decrease. Gradually the data should get larger but it is bound to fluctuate. | 54 bytes |
| Grow Bed Data Point | All daily grow bed data is manually entered here | Date entered daily by STORC technician | The data points collected will be used to monitor the overall efficiency and | Depending on the number of grow beds active, the amount of data may either | 38 bytes |

| | | | | | |
|---|---|---|---|---|---|
| | | | functionality of these stations | increase or decrease. Gradually the data should get larger but it is bound to fluctuate. | |
| Project | This table will store all of the ongoing projects, as well as previous projects that are no longer active. | PI's or webmasters will be entering this information manually for each project. | This is used to help all STORC users keep track of all current and past projects at STORC. | The number of records for projects will fluctuate. Sometimes the number of projects will increase, other times it will decrease. | 8054 bytes |
| Solar Power Data Point | All daily activities related to solar power are recorded in this table | STORC employees will be manually entering in this data as part of their daily duties | This data is used to calculate solar power analitics | This number of records will grow quickly, but overtime will be compressed to make room for new data | 34 bytes |
| Solar Thermal Heat Transferor | This table will contain all data related thermal heat transfer from photovoltaic cells. | Most of this data, will be recorded automatically. Some of the data will have to recorded manually. | This data is used to quantify the functionality of the solar thermal heat transfer. | Depending on the weather and amount of solar panels functioning, the data will fluctuate. | 96 bytes |
| Users | The STORC Employees that will use the system | All users will be manually entered and granted permissions by the administrator | This list will be used authenticate and grant access to the system | The records a will grow and shrink each semester. There will be at most 60 people | 266 megabytes |
| Worm Bin Data Point | This table contains all of the data related to amount of compost produced by worms | STORC employees will be manually entering in this data as part of their daily duties | This data will be used to monitor the amount of worm by product | This data will be input on a regular basis since worms convert garbage into dirt | 34 bytes |

| | | | | | |
|---|---|---|---|---|---|
| | | | produced over time | | |
| Worm Compost Bin | This table is in charge of holding all data related to worm composting. | STORC employees assigned to these stations will have to record some data manually. Other pieces of data will get recorded automatically. | This data will be used to observe functionality and productivity of the worm bins. | This data will be input on a regular basis since worms convert garbage into dirt | 80 bytes |

# APPENDIX C: Coding Standards

Dr. Christensen expressed his interest in another senior project group continuing work on the STORC Dashboard Project after we have concluded our portion of the project. For this reason the coding standard must be very strict. The coding standards are as follows:

1. Each code file will have a header that has the following:
    a. Main Programmer
    b. Other Programmers
    c. File Name
    d. Project Name
    e. References to other files
    f. Date Created
    g. Programming Language
    h. Methods in this file
    i. Description of what this file does
2. Each code file will have a document associated with it (like a Java Doc)
3. All files must have comments that are useful and understandable
    a. Each method must include a comment which summarizes the methods parameters, what this method does, the method's return type, and any other important details that need to be noted
    b. Comments will be scattered throughout the code explaining key functionality and the purpose of the segment of code
        i. Comments will be provided above each loop explaining the purpose of the loop
4. Code must be compatible with different SQL servers if requirements change over time
    a. SQL server names will not be hard coded anywhere within the program
5. In order to reduce confusion and increase maintainability variables and constants will be widely used
    a. If a piece of the code lends itself to be hardcoded we will construct detailed comments and explain why we made that choice
6. All local variables will be made private. This is done for security reasons.

7. All variables must be global in the class unless it is a variable that is only used once in a method ie loops
8. All variables, methods, and class names must be specific to the actions of the descriptor and have a reasonable length (not above 32 characters)
9. No spaghetti code
10. Each piece of the code should not break or modify the action of another piece of code. If this happens the problem needs to be rectified. The person/s responsible for breaking the code are equally responsible for fixing the errors or bugs introduced